TEST PROCEDURES for HIGH LEVEL ARCHITECTURE INTERFACE SPECIFICATION

11 September 1996
VERSION 1.0

Prepared by:

Margaret L. Loper, David W. Roberts
Georgia Tech Research Institute
Georgia Institute of Technology
margaret.loper@gtri.gatech.edu, dw.roberts@gtri.gatech.edu

TABLE OF CONTENTS

1. OVERVIEW

This document contains test procedures for the High Level Architecture Interface Specification, v1.0. The remainder of this section discusses terminology used within this document, organization of the procedures, and how procedures are defined.

1.1 DEFINITIONS

The following definitions apply through this document:

Action Services HLA Interface Services used by the RTI and federate to exchange information in order to accomplish an action.

Black Box Testing A testing process in which the internal design of the Federate Under Test is not visible from the outside.

Conformance Testing The process of verifying that a federate performs in accordance with the interface specification. This is not the same as compliance testing.

Control Services HLA Interface services used by the RTI to initialize federates.

Federate Under Test (FUT) The federate containing the Implementation Under Test.

Initialization Services HLA Interface services used by federates to initialize the RTI.

Management Services HLA Interface Services used to control the operation of the federation or to request information about the state of a federation.

RTI Initialization Data (RID) The data required by the Run Time Infrastructure (RTI) for operation. The required data come from two distinct sources, the Federation Object Model (FOM) product, and the Federation Required Execution Details (FRED).

State Transition Diagrams (STD) A method for specifying the finite state machine of each interface service. The STD is described in terms of state variables.

State Variables A set of variables that describe the state of the federate (or RTI) prior to or as a result of invoking an interface service.

1.2 ORGANIZATION OF TEST PROCEDURES

The services contained in the Interface Specification are at an individual-level, each individual service is described in terms of its inputs, outputs and exceptions. There is only partial guidance for how the individual services are used together or the order in which they should be invoked. The test procedures provide more guidance by creating an execution framework (or recommended order) in which the interface services should be used. This is only guidance; there is no requirement to conform to this framework.

Test procedures are based on both individual-level and functional-level services. For the purpose of these procedures, functional-level services are collections of individual-level services used for a specific function. Individual- and functional -level services are organized into four categories (Initialization, Control, Action, and Management), which form the execution framework, as shown in Figure 1. Each category is described below.

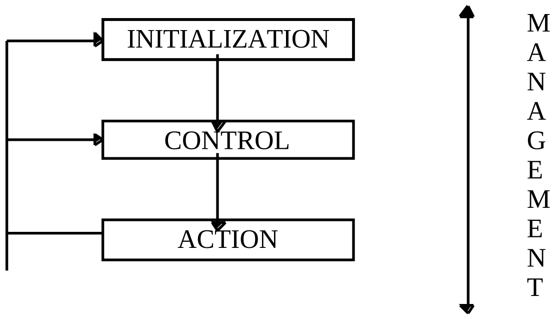

Figure 1: Execution Framework

Initialization Services are used by federates to set state in the RTI. These services typically follow a posting paradigm, in that the federate tells the RTI what it is capable or what it requires for execution. Initialization services are at the individual-level and are shown in Table 1.

Create/Destroy Federation
Join/Resign Federation
Subscribe (Object Class Attribute, Interaction Class)
Publish (Object Class, Interaction Class)
Object Instantiation (Request Id, Register Object)
Request Federation Time
Set Look Ahead
Create Update/Subscription Region, Associate Update Region

Table 1: Initialization Services

Control Services are used by the RTI to initialize federates. These services also follow a posting paradigm in that the RTI tells the federate what publications and interactions it is allowed to update during the execution. Control services are also at an individual-level and are contained in Table 2.

Control Updates
Control Interactions
Discover Object
Change Thresholds

Table 2: Control Services

Action Services are used by the RTI and federate to exchange information in order to accomplish an action. These services follow a request/response paradigm in that the federate requests some action to be completed and the RTI responds with confirmation or notification that the action has taken place. Action services are at the functional-level and are contained in Table 3.

Update/Reflect Attribute
Send/Receive Interaction
Request/Provide Attribute
Modify Region
Change Attribute Transportation Type, Order Type
Change Interaction Transportation Type, Order Type

Delete/Remove Object
Attribute Ownership Acquisition
Attribute Ownership Divestiture
Query Attribute Ownership
Time Advance, Next Event

Table 3: Action Services

Management Services follow a request/response paradigm and are used to control the operation of the federation or to request information about the state of a federation. Management services can be invoked at any time after Initialization; all management services have a pre-condition that a Federation Execution exists and that the Federate is a member of the Execution. Management services are both individual-and functional-level services and are shown in Table 4.

Pause/Resume Federation Execution
Save/Restore Federation Execution
Request Federation Time, Federate Time
Request LBTS, Minimum Next Event Time
Request Lookahead
Flush Queue Request

Table 4: Management Services

1.3 State Transition Diagrams and State Variables

The test procedures contained in this document are based on state transition diagrams (STDs), which is a common method for specifying communication protocols. For each procedure, the individual- or functional-level services are defined in terms of a set of states, the inputs and their effect on the states, and the corresponding outputs. Thus, the state diagrams model the fact that a given input will cause the protocol entity (i.e., the part of the federate that implements the service) to go from one state to another, and possibly also cause a particular output to occur. The state diagrams are present purely for descriptive purposes: they serve as a model of reality.

The procedures in this document include tests for initiating and responding to the interface services. Since it is hard to decouple the federates actions from those of the RTI (in terms of the procedures), the STDs reflect the behavior of both.

In defining the STDs, a set of variables that describe the state of the federate (and RTI) prior to or as a result of invoking an interface service were developed. These variables are called State Variables. The state variables are listed within each STD and are based on the HLA Interface Specification, not the Interface Definition Language (IDL) Application Programmer Interface (API). The State Variables defined for testing are listed in Table 5.

Federation (name, RID, exist/not exist, members)
Federate (name, handle, connection parameters, state, routing space handle)
Pause(pausing/running, label, time)
Save(saving/restoring, label, time, paused, achieved)
Query(request, question, who, result)
Object (class name, id, exist/not exist, attribute name, attribute value, ownership, removal reason, subscription reason handle, update region handle, extents)

Interaction (class name, parameters, id)
Time (federation, federate, logical, tag, lookahead, transport data ordering, handle, #events, offset, scalar, Minimum Next Event Time, LBTS)
ID (requested, number returned, federate limit, federation limit)
Capability (publish, include/exclude flag, subscribe, flag, predicates)
Divest (time, reason, status)
Assumption (time, reason, priority)
Acquisition (time, reason, release time, release reason)

Table 5: State Variables

At this time, there is no standard way to access the state variables for the purpose of testing the Interface Specification. The state variables are presented purely for descriptive purposes.

2. FEDERATE TESTING

This section describes the following issues surrounding federate testing: test set-up, test methods, and performing the test.

2.1 TEST SET-UP

To conduct a test for the interface specification, two federates are required. The first federate is the Federate Under Test (FUT). The FUT is the federate which contains the implementation being tested. The second federate is the "Tester." The term "Tester" is used generically to refer to the person or tool conducting the test. Since there are no test tools currently available for interface testing, the Test Federate supplied with the RTI is considered the "Tester." The FUT and Test Federate are connected via the RTI, as shown in Figure 2.

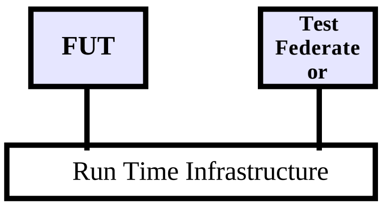

Figure 2: Test Configuration

For Interface testing a federate relies on information in the Simulation Object Model (SOM). This phase of testing does not rely on a federation. However, in order to use the configuration described in Figure 2 and because all interface services (except Create Federation) have a prerequisite that a Federation exists, both the FUT and Test Federate must join a federation. Therefore, for the purpose of federate interface testing the following assumptions are made: the FOM is the same as the SOM and the RID is generated from the SOM.

2.2 TEST METHODS

2.2.1 Black Box Testing

The test set-up described in Section 2.1 is called black box testing. This means that the "Tester" cannot access the internal workings of the implementation of the FUT. Therefore, all the "Tester" can do is invoke services and see how the FUT reacts. This type of testing evaluates the order (and behavior) of services, it does not examine syntax and semantics. Currently, correct syntax is implied from successful interoperation with the RTI. Correct semantics is determined by invoking services according to the execution framework until such time a problem occurs. If a problem is found, the human "Tester" must consider which services are prerequisites for the service that failed and determine what is the likely problem. The black box testing approach described above is a generic process that can be used throughout the test process.

2.2.2 White Box Testing

Another test method that can be used is white box testing. In this approach, the "Tester" has access to internal data from the FUT or can examine the FUT's code during testing. Since tools do not currently exist for this method of testing, a test process cannot be described. At some future date, the state variables described in section 1.3 may be used for observing the internal state of a FUT. At that time a test process describing their use by the "Tester" will be included in this section.

2.3 PERFORMING THE TEST

The first step in performing interface testing is to determine the services to be tested. This is accomplished by documenting a FUT's capabilities in a Service Implementation Statement (SIS). The SIS stands as a record of which services have been tested for each FUT. The SIS is included in Appendix A

The next step of conducting a test is to determine the data to be used for testing. Within the test procedures, supplied parameters are specified as one of four values: supplied/selected by FUT, TBD, selected from RID, or a specific value. Prior to testing, the "Tester" and FUT operator will complete data sheets with the appropriate values for the supplied parameters. During the test, the "Tester" will note the success or failure of each test procedure with respect to the test data. The data sheets used to record the test data are called Service Implementation Logs (SIL). The SIL is a condensed version of the test procedures with a list of the supplied and optional parameters for each service. Separate tests should be conducted for required and optional parameters.

3. ASSUMPTIONS

The following assumptions apply through this document:

1. Federate test procedures presume that the RTI used during testing conforms to the HLA interface specification. Conformance testing of the RTI is outside the scope of this document.
2. Conformance testing excludes any assessment of performance, robustness, or reliability. Only correctness of the implementation is under test, not its speed of operation nor efficiency of its operation, nor the optimization of its code, nor other running capabilities.
3. This documents does not contain procedures for stress testing, invalid or adverse data, or out of bound conditions. These tests are important for conformance testing and should be accomplished operationally.
4. The exception behavior specified in the STD is one example of how a federate can react. There is no requirement to conform to this.
5. Test procedures defined in this document are based solely on the Interface Specification; test procedures are not based on the IDL API.

4. INITIALIZATION AND CONTROL SERVICES

Initialization services are used by federates to set state in the RTI and control (update, interaction) services are used by the RTI to control publication in the federate. These services are shown together in the test procedures since the RTI's control is a response to what the federate initialized.

4.1 CREATE FEDERATION

The Create Federation Execution function is used to build a new federation execution. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		No federation exists	None
Operating State		Wait for services to be issued	None
Event	Federation Managem issues Create Federation Executio	Check Registrations for valid Federation(name, RID); on erro go to operating state	Submit Federation(name, RID) to the RTI
Post Conditions		Register Federation(name)	

4.1.1 Traceability

Section 2.1 Create Federation Execution

4.1.2 Initiating Create Federation Execution

The FUT shall invoke the Create Federation Execution service with the supplied parameters:

Federation Execution Name:	Test
RID:	Selected by FUT

4.2 *DESTROY FEDERATION*

The Destroy Federation Execution function is used to remove the named federation execution from the set of supported federation executions. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Federation exists	Resigned from Federation Execution
Operating State		Wait for services to be issued	None
Event	Federation Manager issues Destroy Federation Executio	Check Registrations for Federation(members, exist); or error go to operating state	Submit Federation(name) to the RTI
Post Conditions		Register Federation(not exist)	

4.2.1 Traceability

Section 2.2 Destroy Federation Execution

4.2.2 Initiating Destroy Federation Execution

The FUT shall invoke the Destroy Federation Execution service with the supplied parameters:

Federation Execution Name:	Test

4.3 *JOIN FEDERATION*

The Join Federation Execution function is used to affiliate the federate with the federation execution and declare important properties about the federate. The STD is below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows members of federation	None
Operating State		Knows services issued	None
Event	Federate issues Join Federation Execution	Check Registrations for valid Federate(name) and Federation(name, members); o error go to operating state	Submit Federate(name) Federation(name), and optional connection parameters to the RTI
Post Conditions		Register Federation(name); Return Federate (handle) to federate	Federate is member of named Federation Executio

4.3.1 Traceability

Section 2.3 Join Federation Execution

4.3.2 Initiating Join Federation

The FUT shall invoke the Join Federation Execution service with the supplied parameters:

Federate Name:	supplied by FUT
Federation Execution Name:	Test
Connection Parameters:	tbd/optional

The RTI shall return the parameters:

Federate Handle	*selected by RTI*

4.4 Resign Federation

The Resign Federation Execution function is used to cease the participation of the federate in the named federation execution. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership	Member of Federation
Operating State		Knows services issued	Knows current state
Event	Federate/Federation Manager issues Resign Federation Execution	Check Registrations for valid Federate(name) and Federation(member); perform optional directive if necessary; on error go to operating state	Submit Federate(name), Federation (name) and optional directive to the RTI
Post Conditions		Remove Federate(name) from Federation(member)	Federate not a member of Federation Execution

4.4.1 Traceability

Section 2.4 Resign Federation Execution

4.4.2 Initiating Resign Federation Execution

The FUT shall invoke the Resign Federation service with the supplied parameters:

Federate name	supplied by FUT
Federation execution name:	Test

4.4.3 Initiating Resign Federation Execution (Optional Parameters)

The FUT shall invoke the Resign Federation service with the supplied parameters:

Federate name	supplied by FUT
Federation execution name:	Test
Optional directive:	(1) release all attribute ownership, (2) delete all objects for which the federate holds delete privilege, or (3) perform action (2) and then action (1)

4.5 Subscribe Object Class Attribute

The Subscribe Object Class Attribute function is used to control which classes of objects the federate will discover and which attribute value the federate will receive from that class.

This function is to be called iteratively by the FUT to subscribe to multiple attributes of an object class. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Knows current state
Event	Federate issues Subscribe Object Class Attribute	Check Registrations for valid Federate(name) and Federation(member); check RID for valid Object(class, attribute name), Capability(include/unsubscribe ; verify object is associated wit region; on error go to operating state	Submit Federate(name), Federation(name), Object(class name, attribut name), Capability(include/exclude flag, region), Federate(name), and Federation(name) to RTI
Post Conditions		Register Capability(subscription, include/unsubscribe) and Optional(region) for Obect(attribute name)	

4.5.1 Traceability

Section 3.3 Subscribe Object Class Attribute

4.5.2 Initiating Subscribe Object Class Attribute

The FUT shall invoke the Subscribe Object Class Attribute service with the supplied parameters:

Object Class Name:	selected from RID
Include/Exclude Option:	selected by FUT
if Include,	
Attribute Name:	selected from RID
Federate Handle:	returned from RTI
Federation Execution Name:	Test

4.5.3 Initiating Subscribe Object Class Attribute (Optional Parameters)

The FUT shall invoke the Subscribe Object Class Attibute service with the supplied parameters:

Object Class Name:	selected from RID
Include/Unsubscribe Option:	selected by FUT
if Include,	
Attribute Names:	selected from RID
Federate Handle:	returned from RTI
Federation Execution Name:	Test
Region:	optional

4.6 *Subscribe Interaction Class*

The Subscribe Interaction Class function is used to specify the classes of interactions which should or should not be sent to the federate. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Knows current state
Event	Federate issues Subscribe Interaction Class	Check Registrations for valid Federate(name) and Federation(member); check RID for valid Interaction(class name); on error go to operating state	Submit Federate(name), Federation(name), Interaction(class name), Capability(sub, include/exclude flag) to the RTI
Post Conditions		Register Capability(subscribe, include/exclude flag) for Interaction(class)	

4.6.1 Traceability

Section 3.4 Subscribe Interaction Class

4.6.2 Initiating Subscribe Interaction Class

The FUT shall invoke the Subscribe Interaction class service with the supplied parameters:

Interaction Class Name:	selected from RID
Include/Exclude Flag:	selected by FUT (include,exclude)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

4.6.3 Initiating Subscribe Interaction Class (Optional Parameters)

The FUT shall invoke the Subscribe Interaction class service with the supplied parameters:

Interaction Class Name:	selected from RID
Include/Exclude Flag:	selected by FUT (include,exclude)
Federate Handle:	returned from RTI
Federation Execution Name:	Test
Region:	optional

4.7 *Publish Object Class*

The Publish Object Class function is used to indicate which classes of objects the federate is capable of providing to the federation and specifies the attributes to be supplied for those object classes. The Control Updates function is used to tell the federate that the specified attributes for the specified class or object are or are not required somewhere in the federation. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Knows current state
Event 1	Federate issues Publish Object Class	Check Registrations for valid Federate(name) and Federation(member); check RID for valid Object(class name, attribute names); on erro go to operating state	Submit Federate(name), Federation(name), Object(class name, attribute names), Capability(publish, include/exclude flag) to RT
Post Conditions		Register Capability(publish, include/exclude flag) for Object(attribute names)	
Event 2	RTI issues Control Updates	Submit Object(class, attribute names) and Capability(update/no update) t the federate	Check Registrations for valid published Object(attributes); on error go to operating state
Post Conditions			Register Capability(update/no update) for Object(attribute names)

4.7.1 Traceability

Section 3.1 Publish Object Class
Section 3.5 Control Updates

4.7.2 Initiating Publish Object Class

The FUT shall invoke the Publish Object Class service with the supplied parameters:

Object Class Name:	selected from RID
Attribute Names:	selected from RID
Include/Exclude Flag:	selected by FUT (include,exclude)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Control Updates service to the FUT with the supplied parameters:

Object Class:	*supplied by RTI*
Attribute Names:	*supplied by RTI*
Update Flag:	*selected by RTI (update, no update)*
Federate Handle:	selected by RTI
Federation Execution Name:	Test

4.8 PUBLISH INTERACTION CLASS

The Publish Interaction Class function is used to specify which classes of interactions the federate will be publishing to the federation. The Control Interactions function is used to tell the federate that the specified class of interactions is or is not required somewhere in the federation. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Knows current state
Event 1	Federate issues Publish Interaction Class	Check Registrations for valid Federate(name) and Federation(member); check RID for valid Interaction(class) on error go to operating state	Submit Federate(name), Federation(name), Interaction(class name), Capability(publish, include/exclude flag) to the RTI
Post Conditions		Register Capability(publish, include/exclude flag) for Interaction(class name)	
Event 2	RTI issues Control Interactions	Submit Interaction(class name) Capability(send/no send flag) t the federate.	Check Registrations for valid published Interaction(class name); on error go to operating state
Post Condition			Register Capability(send/no send) for Interaction(class name)

4.8.1 Traceability

Section 3.2 Publish Interaction Class
Section 3.6 Control Interactions

4.8.2 Initiating Publish Interaction Class

The FUT shall invoke the Publish Interaction Class service with the supplied parameters:

Interaction Class Name:	selected from RID
Include/Exclude Flag:	selected by FUT (include, exclude)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Control Interactions service to the FUT with the supplied parameters:

Interaction Class Name:	*supplied by FUT*
Publish Flag:	*selected by RTI (publish, not publish)*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

4.9 REQUEST ID

The Request Id function is used to request federation execution-unique object ID numbers. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Knows current state
Event	Federate issues ID Request	Check Registrations for valid Federate(name), Federation(memb and number of IDs that are not reused or reserved; on error go to operating state	Submit Federate(handle), Federation(name), ID(requestec to the RTI
Post Conditions		Return ID(numbers) to the federate	

4.9.1 Traceability

Section 4.1 Request Id

4.9.2 Initiating Request Id

The FUT shall invoke the Request Id service with the supplied parameters:

Number of Ids:	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Set of Ids:	*selected by RTI*

4.10 REGISTER OBJECT

The Register Object function is used to link an object Id with an instance of an object class. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Publishing object class
Event	Federate issues Instantiate Object	Check RID for valid Object(class name); check Registrations for valid Federate(name), Federation(member), object(id Capability (publish, include/exclude flag); on error go to operating state.	Submit Object(class name, id), to the RTI
Post Conditions		Register Object(id, ownership)	Register Object(id, ownership)

4.10.1 Traceability

Section 4.2 Register Object

4.10.2 Initiating Register Object

The FUT shall invoke the Register Object service with the supplied parameters:

Object Class Name:	selected from RTI
Object ID:	returned from RTI in 4.8
Federate Handle:	returned from RTI
Federation Execution Name:	Test

4.11 *Set Lookahead*

The Set Look Ahead function is used to set the desired value of the federate's lookahead. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	
Event	Federate issues Set Lookahead	Check Registrations for Federate(name), Federation(member), and for valid Time(lookahead); on error go to operating state	Submit Federate(name), Federation(name), Time(lookahead) to RTI
Post Conditions		Register Time(lookahead) for Federate(name)	

4.11.1 Traceability

Section 6.5 Set Lookahead

4.11.2 Initiating Set Lookahead

The FUT shall invoke the Set Look Ahead service with the supplied parameters:

Lookahead:	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

4.12 *Create Update Region*

The Create Update Region function is used to create a subset of the specified routing space. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	
Event	Federate issues Create Update Region	Check Registrations for valid Federate(name), Federation(member), check RID for valid Federate(Routing Space Handle), Object(Extents on error go to operating state.	Submit Federate(name,Routing Space Handle, Extents), Federation(name) to the RT
Post Conditions		Return Object(Update Region Handle)	Register Federate(Routing Space Handle, Extents), Object(Update Region Handle)

4.12.1 Traceability

Section 7.1 Create Update Region

4.12.2 Initiating Create Update Region

The FUT shall invoke the Create Update Region service with the supplied parameters:

Routing Space Handle	Selected from RID
Extents	Selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Update Region Handle:	*selected by RTI*

4.13 *CREATE SUBSCRIPTION REGION*

The Create Subscription Region function is used to create a subset of the specified routing space. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	
Event	Federate issues Create Update Region	Check check Registrations for valid Federate(name), Federation(member), check RID for valid Federate(Routing Space Handle, Extents); on error go to operating state.	Submit Federate(name, Routing Space Handle), Federation(Name), Object(Extents) to the RTI
Post Conditions		Return Object(Subscription Region Handle)	Register Federate(Routing Space Handle), Object(Extents, Subscriptio Region Handle)

4.13.1 Traceability

Section 7.1 Create Subscription Region

4.13.2 Initiating Create Subscription Region

The FUT shall invoke the Create Subscription Region service with the supplied parameters:

Routing Space Handle	Selected from RID
Extents	Selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Subscription Region Handle:	*selected by RTI*

4.14 *Associate Update Region*

The Associate Update Region service associates an update region with attributes of a specific object or an interaction class. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Object(Id, Attributes) is registered, and being published, and update region exists.
Event	Federate issues Associate Update Region	Check Registrations for valid Federate(name), Federation(member), object(id update region handle), Capability (publish, include/exclude flag); on error go to operating state.	Submit Federate(name), Federation(name), Object(update region handle, association instruction, Id, Attributes) t the RTI
Post Conditions		Register Object(id, update region)	Register Object(id, ownership)

4.14.1 Traceability

Section 7.3 Associate Update Region

4.14.2 Initiating Associate Update Region for Attributes of an Object

The FUT shall invoke the Associate Update Region service with the supplied parameters:

Update Region Handle	returned from RTI in 4.11
Object ID:	returned from RTI in 4.8
Attributes:	selected by FUT
Association Instruction	(form, break)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

4.14.3 Initiating Associate Update Region for Interaction Classes

The FUT shall invoke the Associate Update Region service with the supplied parameters:

Update Region Handle	returned from RTI in 4.11
Interaction Class:	selected by FUT
Association Instruction	(form, break)
Federate Handle:	returned from RTI
Federation Execution Name:	Test
Federation Execution Name:	Test

4.15 *Delete Region*

The Delete Region function is used delete the specified update or subscription region. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Update/subscription region exists.
Event	Federate issues Delete Region	Check Registrations for valid Federate(name), Federation(member), Object(Update/Subscription Handle); on error go to operating state.	Submit Object(Update/Subscription Handle) to the RTI
Post Conditions		Delete Object(Update/Subscription Region)	

4.15.1 Traceability
Section 7.6 Delete Region

4.15.2 Initiating Delete Region
The FUT shall invoke the Delete Region service with the supplied parameters:

Update/Subscription Region Handle	returned from RTI in 4.11
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5. ACTION AND CONTROL SERVICES

Action services are used by the federate and RTI to exchange information in order to accomplish an action for the federate. The control service in this section is used by the RTI to inform the federate of an object which requires reflection.

5.1 *DISCOVER OBJECT*

The Discover Object function is used to inform the federate that the RTI has discovered an object in the federation when the following occur: 1) An attribute update is received from another federate, 2)The federate has neither registered nor discovered the object, 3) The update satisfies the federate's description, and 4) The update is in the associated region (if regions are being used.) This service is issued by the RTI as a response to the Update Attribute Value services. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STAT
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation	Member of Federation
Operating State		Knows publication and subscription capabilities of federates; waiting for services to be issued	Publishing object class	Subscribed to object class
Event 1	Federate A issues Update Attribute Values	Check RID for valid (attribu name, attribute value); chec Registrations for valid Object(id, ownership); check for valid Time(federation,tag on error go to operating state	Submit Federate(name Federation(Name), Object(id, attribute name, attribute value), Time(federation, tag) t the RTI	
Post Conditions		Return Time(handle) to federate; Search Registrations for valid federate(name), federation(member), Object(id, attribute name, attribute value) with Capability(subscribe, include/exclude flag)		
Event 2	RTI issues Discover Object to Federate B	Submit Object(class, id), Time(federation,handle, tag) to the federate		Check Registrations fo valid subscribed Object(id); check for valid Time(federation, tag); on error go to operating state
Post Conditions			Register reflected Object(id)	

5.1.1 Traceability

Section 4.3 Update Attribute Values

Section 4.4 Discover Object

5.1.2 Responding to Discover Object

The Test Federate shall invoke the Update Attribute Values service with the supplied parameters:

Object Id:	returned from RTI in 4.8
Attribute Name:	selected from RID
Attribute Value:	selected by Test Federate
Federation Time:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Event Retraction Handle:	*selected by RTI*

The RTI shall invoke the Discover Object service to the FUT with the supplied parameters:

Object ID:	*supplied by Test Federate*
Object Class Name:	*selected by RTI*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.2 *UPDATE/REFLECT ATTRIBUTES*

The Update/Reflect Attributes functions are used to provide the current attribute values to the federation. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STAT
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation	Member of Federation
Operating State		Wait for services to be issued	Publishes object attributes	Subscribes to object attributes
Event 1	Federate A issues Update Attribute Values	Check RID for valid (attribute name, attribute value);, checl Registrations for valid Federate(name), Federation(member), Object(id, ownership); check for valid Time(federate, tag) o error go to operating state	Submit Federate(name), Federate(name), Object(id, attribute name attribute value), Time(federate, tag) to th RTI	
Post Conditions		Return Time(handle) t federate; Search Registrations for valid federate(name), federation(member), Object(id, attribute name, attribute value) with Capability(subscribe, include/exclude flag)		
Event 2	RTI issues Reflect Attribute Values to Federate B	Submit Object(id, attribute name, attribu value) and Time(federation, handl to federates registered for Capability(sub, inc/exc flag)		Check Registrations fo valid Object(id); checl RID for valid Object(attribute name, attribute value); check for valid Time(federation, timestamp); on error g to operating state
Post Conditions				

5.2.1 Traceability

Section 4.3 Update Attribute Values

Section 4.5 Reflect Attribute Values

5.2.2 Initiating Update Attribute Values

The FUT shall invoke the Update Attribute Values service with the supplied parameters:

Object Id:	returned from RTI in 4.8
Attribute Names, Values	selected by FUT
Federate Time:	selected by FUT
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Event Retraction Handle:	*selected by RTI*

The RTI shall invoke the Reflect Attribute Values service to the Test Federate with the supplied parameters:

Object Id:	*selected by FUT*
Attribute Names, Values:	*selected by FUT*
Federation Time:	*selected by FUT*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.2.3 Responding to Update Attribute Values

The Test Federate shall invoke the Update Attribute Value service with the supplied parameters:

Object Id:	returned from RTI in 4.8
Attribute Names, Values:	selected by Test Federate
Federate Time:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Event Retraction Handle:	*selected by RTI*

The RTI shall invoke the Reflect Attribute Value service to the FUT with the supplied parameters:

Object Id:	*supplied by Test Federate*
Attribute Names, Values:	*supplied by Test Federate*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.3 *SEND/RECEIVE INTERACTION*

The Send/Receive Interaction function is used to inform the federation of an action taken by one object towards another object. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation and publishing interaction class	
Operating State		Knows publication and subscription capabilitie of federates; waiting for services to be issued.	Publishing Interaction Class	Subscribing to Interaction Class
Event 1	Federate A issues Send Interaction	Check RID for valid Interaction(class name parameters); check Registrations for federate(name), federation(member); check for valid Time(federation); on error go to operating state	Submit Interaction(class name, parameters), and Time(federation,tag) to RTI	
Post Conditions		RTI returns Time(handle) to federate		
Event 2	RTI issues Receive Interaction to Federate B	Submit Interaction(class, parameters), and Time(federation,tag) t the federate		Interaction(class, parameters); check Registrations for valid subscription Interaction(class); chec for valid Time(federation); on error go to operating
Post Conditions				

5.3.1 Traceability

Section 4.6 Send Interaction
Section 4.7 Receive Interaction

5.3.2 Initiating Send Interaction

The FUT shall invoke the Send Interaction service with the supplied parameters:

Interaction Class Name:	selected from RID

Parameter Names, Values:	selected from RID
Federation Time:	selected by FUT
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Receive Interaction service to the Test Federate with the parameters:

Interaction Class Name:	*supplied by FUT*
Interaction Parameter Names, Values:	*supplied by FUT*
Federation Time:	*supplied by FUT*
User Supplied Tag	*selected by FUT*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.3.3 Responding to Send Interaction

The Test Federate shall invoke the Send Interaction service with the supplied parameters:

Interaction Class Name:	selected from RID
Parameter Names, Values:	selected from RID
Federate Time:	selected by Test Federate
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Receive Interaction service to the FUT with the parameters:

Interaction Class Name:	*supplied by Test Federate*
Interaction Parameter Names, Values:	*supplied by Test Federate*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.4 Request/Provide Attribute

The Request/Provide Attribute function is used to request the current attribute values for the specified object or object class from the federation member responsible for publishing this information. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation membership and initialization content	Member of Federation	Member of Federation
Operating State		Wait for services to be issued	Subscribing to object attri	Publishing object attribut
Event 1	Federate A issues Request Attribute Value Update	Check Registrations for federate(n federation(member), valid Object(check RID for valid Object(attribu name); on error go to operating sta	Submit Federate(name), Federation(name), Object attribute name) to the RTI	
Post Conditions		Check Registrations for Object(id) Federate(id) to locate federate pub attributes		
Event 2	RTI issues Provid Attribute Value Update to Federat B	Submit Object(id, attribute name) federate		Check RID for valid Object(attribute name); c Registrations for valid Object(id); on error go to operating state
Post Conditions				
Event 3	Federate B issues Update Attribute Values	Check RID for valid (attribute nam attribute value); check Registratio federate(name), federation(membe valid Object(id, ownership); check valid Time(federation); on error g operating state		Submit Federate(name), Federation(name), Objec attribute name, attribute and Time(federation, tag the RTI
Post Conditions		Return Time(handle) to federate B Search Registions for Object(id, at name, attribute value) with Capability(subscribe, include/excl		
Event 4	RTI issues Reflec Attribute Values	Submit Object(id, attribute name, value) and Time(federation, tag) t federates Registered for Capability(subscribe, include/excl	Check Registrations for va Object(id); check RID for Object(attribute name, att value); check for valid Time(federation, tag); on go to operating state	
Post Conditions				

5.4.1 Traceability

Section 4.14 Request Attribute Value Update
Section 4.15 Provide Attribute Value Update
Section 4.3 Update Attribute Values
Section 4.5 Reflect Attribute Values

5.4.2 Initiating Request Attribute Value

The FUT shall invoke the Request Attribute Value Update service with the supplied parameters:

Object Id or Object Class	selected by FUT
Attribute Names:	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Provide Attribute Value Update service to the Test Federate with the parameters:

Object Id:	*supplied by FUT or RTI*
Attribute Names:	*supplied by FUT*
Federate Handle:	*supplied by FUT*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Update Attribute Values service with the supplied parameters:

Object Id:	selected by FUT or RTI
Attribute Name:	selected by FUT
Attribute Value:	selected by Test Federate
Federate Time:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Event Retraction Handle:	*selected by RTI*

The RTI shall invoke the Reflect Attribute Values service to the FUT with the supplied parameters:

Object Id:	*selected by FUT or RTI*
Attribute Names, Values	*selected by Test Federate*
Federation Time:	*selected by Test Federate*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.4.3 Responding to Request Attribute Value

The Test Federate shall invoke the Request Attribute Value Update service with the supplied parameters:

Object Id or Object Class:	selected by Test Federate
Attribute Names:	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Provide Attribute Value Update service to the FUT with the parameters:

Object Id:	*selected by Test Federate*

Attribute Names:	*selected by Test Federate*
Federate Handle:	*supplied by Test federate*
Federation Execution Name:	*Test*

The FUT shall invoke the Update Attribute Values service with the supplied parameters:

Object Id:	selected by Test Federate or RTI
Attribute Names, Values:	selected by FUT
Federation Time:	selected by FUT
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameter:

Event Retraction Handle:	*selected by RTI*

The RTI shall invoke the Reflect Attribute Value service to the Test Federate with the supplied parameters:

Object Id:	*selected by Test Federate or RTI*
Attribute Names, Values:	*selected by FUT*
Federation Time:	*supplied by FUT*
User Supplied Tag	*selected by FUT*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

5.5 *CHANGE ATTRIBUTE TRANSPORTATION TYPE*

The Change Attribute Transportation Type service is used to change the transportation type for the specified attributes on the specified object.

	SERVICE	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Owns attribute(s)
Event 1	Federate issues Change Attribute Transportation Type	Check for Federate(name), Federate(member); check Registrations for ownership Object(Id, Attribute Set), che for valid Time(transport service); on error go to operating state	Submit Federate(name), Federation(name), Object(Id Attribute Set), Time(transportation service) to RTI,
Post Conditions		Register Time(transportation service)	Register Time(transportation service)

5.5.1 Traceability

Section 4.10 Change Attribute Transportation Type

5.5.2 Initiating Change Attribute Transportation Type

The FUT shall invoke the Change Attribute Transportation Type service with the supplied parameters:

Object Id	selected from RID
Attribute Names	set of attributes of object
Transportation Type	tbd
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5.6 *Change Attribute Order type*

The Change Attribute Order Type service is used to change the data ordering type for the specified attributes on the specified object.

	SERVICE	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Owns attribute(s)
Event 1	Federate issues Change Attribute Order Type	Check for Federate(name), Federate(member); check Registrations for ownership Object(Id,Attribute Set), Time(Data Ordering Type); or error go to operating state	Submit Federate(name), Federation(name), Object(Id Attribute Set) Time(Data Ordering Type) to RTI,
Post Conditions		Register Time(Data Ordering Type)	Register Time(Data Ordering Type)

5.6.1 Traceability

Section 4.11 Change Attribute Order Type

5.6.2 Initiating Change Attribute Order Type

The FUT shall invoke the Change Attribute Order Type service with the supplied parameters:

Object Id	selected from RID
Attribute Names	set of attributes of object
Data Ordering Type	tbd
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5.7 *Change Thresholds*

The Change Thresholds service provides the federate with the new values of the thresholds for each of the dimensions of a routing space. The new threshold data should be used to decide when to use the Modify Region service. The Modify Region service is used to change the bounds of the update or subscription to reflect the specified set of extents. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Wait for services to be issued	Object(Id, Attributes) is registered, and being published, and update region exists.
Event	Federate issues Associate Update Region	Check RID for valid Object(class); check Registrations for valid Federate(name), Federation(member), object(id update region handle), Capability (publish, include/exclude flag); on error go to operating state.	Submit Federate(name), Object(update region handle, association instruction, Id, Attributes) t the RTI
Post Conditions		Register Object(id, update region)	Register Object(id, ownership)

5.7.1 Traceability

Section 7.4 Change Thresholds
Section 7.5 Modify Region

5.7.2 Initiating Change Thresholds

The RTI shall invoke the Change Thresholds service with the supplied parameters:

Update/Subscription Region Handle	*selected by RTI*
Thresholds	*selected by RTI*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Modify Region service with the supplied parameters:

Update/Subscription Region Handle	returned from RTI in 4.11
Extents	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5.8 CHANGE INTERACTION TRANSPORTATION TYPE

The Change Interaction Transportation Type service is used to change the transportation type for the specified interaction class.

	SERVICE	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Publishing interaction
Event 1	Federation Manager issues Change Federate Time	Check for Federate(name), Federate(member); check Registrations for publishing of Interaction(class), Time(Transportation Type); on error go to operating state	Submit Federate(name), Federation(name), Interaction(class) Time(Transportation Type) to RTI.
Post Conditions		Register Time(Transportation Type) for Federate(name)	Register Time(Transportation Type)

5.8.1 Traceability

Section 4.12 Change Interaction Transportation Type

5.8.2 Initiating Change Interaction Transportation Type

The FUT shall invoke the Change Interaction Transportation Type service with the supplied parameters:

Interaction Class	selected from RID
Transportation Type	tbd
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5.9 *CHANGE INTERACTION ORDER TYPE*

The Change Interaction Order Type service is used to change the data ordering type for the specified Interaction class.

	SERVICE	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Publishing interaction
Event 1	Federation Manager issues Change Federate Time	Check for Federate(name), Federate(member); check Registrations for publishing of Interaction(class name), Time(Data Ordering Type); or error go to operating state	Submit Federate(name), Federation(name), Interaction(class name), Time(Data Ordering Type) to RTI.
Post Conditions		Register Interaction(Data Ordering Type) for Federate(name);	Register Interaction(Data Ordering Type)

5.9.1 Traceability

Section 4.13 Change Interaction Order Type

5.9.2 Initiating Change Interaction Order Type

The FUT shall invoke the Change Interaction Order Type service with the supplied parameters:

Interaction Class	selected from RID
Data Ordering Type	tbd
Federate Handle:	returned from RTI
Federation Execution Name:	Test

5.10 DELETE/REMOVE OBJECT

The Delete/Remove Object function is used to inform the federation that an object is to be removed from the federation. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation	Member of Federation
Operating State		Knows publication and subscription capabilities of federates; waiting for services to be issued	Owns the object ID attribute	Reflects the object ID attribute
Event 1	Federate A issues Delete Object	Check Registrations for valid Federate(name), Federation(member), Object(id, ownership); check for valid Time(federation, tag); on error go to operating state	Submit Federate(name), Federation(name), Object(id) and Time(federation, tag) to the federate	
Post Conditions		Return Time(handle); Register Object(not exist)	Register Capability(publish, include/exclude flag) for Object(id)	
Event 2	RTI issues Remove Object to Federate B	Submit Object(id,removalreason) and Time(federation, tag) to federates subscribed to Object(id)		Check Registrations for valid Object(id); check for valid Time(federation); on error go to operating state
Post Conditions				Register Object(not exist)

5.10.1 Traceability

Section 4.8 Delete Object
Section 4.9 Remove Object

5.10.2 Initiating Delete Object

The FUT shall invoke the Delete Object service with the supplied parameters:

Object Id:	selected by FUT
Federation Time:	selected by FUT
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the following parameter:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Remove Object service to the Test Federate with the supplied parameters:

Object Id:	*supplied by FUT*
Object Removal Reason:	*(deleted, out-of-scope)*
Federation Time:	*supplied by FUT*
User Supplied Tag	*selected by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.10.3 Initiating Delete Object (Optional Parameters)

The FUT shall invoke the Delete Object service with the supplied parameters:

Object Id:	selected by FUT
Federation Time:	selected by FUT
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the following parameter:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Remove Object service to the Test Federate with the supplied parameters:

Object Id:	*supplied by FUT*
Object Removal Reason:	*(deleted, out-of-scope)*
Federation Time:	*supplied by FUT*
User Supplied Tag	*selected by FUT*
Retraction Handle:	*supplied by FUT/optional*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.10.4 Responding to Delete Object

The Test Federate shall invoke the Delete Object service with the supplied parameters:

Object Id:	selected by Test Federate
Federation Time:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the following parameter:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Remove Object service to the FUT with the supplied parameters:

Object Id:	*supplied by Test Federate*
Object Removal Reason:	*(deleted, out-of-scope)*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*selected by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.10.5 Responding to Delete Object (Optional Parameters)

The Test Federate shall invoke the Delete Object service with the supplied parameters:

Object Id:	selected by Test Federate
Federation Time:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the following parameter:

Retraction Handle:	*selected by RTI*

The RTI shall invoke the Remove Object service to the FUT with the supplied parameters:

Object Id:	*supplied by Test Federate*
Object Removal Reason:	*(deleted, out-of-scope)*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*selected by Test Federate*
Retraction Handle:	*supplied by Test Federate/optional*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.11 ATTRIBUTE OWNERSHIP ACQUISITION

The Attribute Ownership Acquisition function is used to request privilege to own the specified attributes of the specified objects. The RTI will request that the federate owning the attributes release ownership, and then notify the initiating federate of the result. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation membe and RTI initialization co	Member of Federation	Member of Federation
Operating State		Waiting for services to b issued	Invoked Publish Object Cl Publish Object Attributes; Owns specified object attr	Invoked Publish Object Cl Publish Object Attributes
Event 1	Federate B issues Rec Attribute Ownership Acquisition	Check Registrations for v Object(id, attribute name check for valid Federation(member), Federate(name) for Capability(publish); che Acquisition(time) >= Time(federation); check valid time; on error go to operating state		Submit Federation(name) Federate(name), Object(i attribute names), Time(ta RTI
Post Conditions		Search Registrations for Federate(name) of objec ownership, attribute nam		
Event 2	RTI issues Request Attribute Ownership Release to Federate A	Submit Object(id, attribu names) and Time(tag) to federate	Check Registrations for va Object(id, attribute names error go to operating state	
Post Conditions		Register Object (id, attrib names, ownership) for Federate(null)	Return Object(id, attribute names); Register release o Object (id, attribute name ownership)	
Event 3	RTI issues Attribute Ownership Acquisitio Notification to Federa	Submit Object(id, attribu names) to federate		Check Registrations for va Object(id, attribute names error go to operating state
Post Conditions		Register Object (id, owne attribute names) for Federate(id)		Register Object (id, owne attribute names)

5.11.1 Traceability

Section 5.5 Request Attribute Ownership Acquisition
Section 5.6 Request Attribute Ownership Release
Section 5.4 Attribute Ownership Acquisition Notification

5.11.2 Initiating Attribute Ownership Acquisition

The FUT shall invoke the Request Attribute Ownership Acquisition service with the supplied parameters:

Object Id:	selected by FUT
Attribute Names:	selected by FUT
User Supplied Tag	selected by FUT

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Release to Test Federate with the parameters:

Object Id:	*selected by FUT*
Attribute Names	*selected by FUT*
User Supplied Tag	*selected by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall return the parameters:

Attribute Names:	selected by Test Federate

The RTI shall invoke the Attribute Ownership Acquisition Notification to the FUT with parameters:

Object Id:	*selected by FUT*
Attribute Names:	*returned from Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.11.3 Responding to Attribute Ownership Acquisition

The Test Federate shall invoke the Request Attribute Ownership Acquisition service with the supplied parameters:

Object Id:	selected by Test Federate
Attribute Names:	selected by Test Federate
User Supplied Tag	selected by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Release to the FUT with the parameters:

Object Id:	*selected by Test Federate*
Attribute Names:	*selected by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall return the parameters:

Attribute Names:	selected by FUT

The RTI shall invoke the Attribute Ownership Acquisition Notification to the Test Federate with parameters:

Object Id:	*selected by Test Federate*
Attribute Names:	*returned from FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.12 *Attribute Ownership Divestiture*

The Attribute Ownership Divestiture function is used to tell the RTI that the federate no longer wants to own the specified attributes of the specified objects. The RTI will look for a new owner for the attributes and notify the initiating and responding federates of the result. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation members and RTI initialization cont	Member of Federation	Member of Federation
Operating State		Wait for services to be issued	Owns specified attribute object IDs	Invoked Publish Object C or Publish Object Attribut
Event 1	Federate A issues Request Attribute Ownership Divestit	Check Registrations for va Federate(name), Federation(member), Obje and Object (id, ownership) check RID for valid Object(attribute names); o error go to operating state	Submit Federation(name Federation(name), Obje attribute names, divestit condition), Time(tag), o Federate(names) RTI	
Post Conditions		If Federate(name) not spec check Registrations for federates with Capability(publish)		
Event 2	RTI issues Request Attribute Ownershi Assumption to Federate B	Submit Object(id, attribute names),		Check Registrations for v Object(id, attribute names error go to operating state
Post Conditions				Return Object(attribute na
Event 3	RTI issues Attribute Ownership Acquisition Notification to Federate B	Submit Object(id, attribute names) to federate		Check Registrations for v Object(id, attribute names error go to operating state
Post Conditions				Register Object (id, owne attribute name)
Event 4	RTI issues Attribut Ownership Divestit Notification to Federate A	Submit Object(ids, attribut names) federate	Check Registrations for Object(id, attribute nam ownership); on error go t operating state.	
Post Conditions		Register Object (id, owner attribute name) with Federate(id)	Register release of Obje ownership, attribute nam	

5.12.1 Traceability

Section 5.1 Request Attribute Ownership Divestiture
Section 5.2 Request Attribute Ownership Assumption
Section 5.4 Attribute Ownership Acquisition Notification
Section 5.3 Attribute Ownership Divestiture Notification

5.12.2 Initiating Attribute Ownership Divestiture

The FUT shall invoke the Request Attribute Ownership Divestiture service with the supplied parameters:

Object Id:	selected by FUT
Attribute Names:	selected by FUT
Ownership Divestiture Condition:	negotiated
User Supplied Tag	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Assumption to the Test Federate service with the supplied parameters:

Object Id:	*supplied by FUT*
Attribute Names:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall return the parameters:

Attribute Names:	selected from those supplied by RTI

The RTI shall invoke the Attribute Ownership Acquisition Notification to Test Federate with the supplied parameters:

Object Id:	*supplied by Test Federate*
Attribute Names:	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Attribute Ownership Divestiture Notification to the FUT with the supplied parameters:

Object Ids:	*supplied by Test Federate*
Attribute Names:	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.12.3 Initiating Attribute Ownership Divestiture (Optional Parameters)

The FUT shall invoke the Request Attribute Ownership Divestiture service with the supplied parameters:

Object Id:	selected by FUT
Attribute Names:	selected by FUT
Ownership Divestiture Condition:	unconditional
User Supplied Tag	selected by FUT
Federate Name:	Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Assumption service to the Test Federate with the supplied parameters:

Object Id:	*supplied by FUT*
Attribute Names:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall return the parameters:

Attribute Names:	selected from those supplied by RTI

The RTI shall invoke the Attribute Ownership Acquisition Notification to the Test Federate with the supplied parameters:

Object Id:	*supplied by Test Federate*
Attribute Names:	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Attribute Ownership Divestiture Notification service to the FUT with the supplied parameters:

Object Ids:	*supplied by Test Federate*
Attribute Names:	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.12.4 Responding to Attribute Ownership Divestiture

The Test Federate shall invoke the Request Attribute Ownership Divestiture service with the supplied parameters:

Object Id:	selected by Test Federate
Attribute Names:	selected by Test Federate
Ownership Divestiture Condition:	negotiated
User Supplied Tag	selected by Test Federate

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Assumption service to the FUT with the supplied parameters:

Object Id:	*supplied by Test Federate*
Attribute Names	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall return the parameters:

Attribute Name:	selected from those supplied by RTI

The RTI shall invoke the Attribute Ownership Acquisition Notification to the FUT with the supplied parameters:

Object Id:	*supplied by FUT*
Attribute Names:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Attribute Ownership Divestiture Notification to Test Federate service with the supplied parameters:

Object Ids:	*supplied by FUT*
Attribute Name:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.12.5 Responding to Attribute Ownership Divestiture (Optional Parameters)

The Test Federate shall invoke the Request Attribute Ownership Divestiture service with the supplied parameters:

Object Id:	selected by Test Federate
Attribute Names:	selected by Test Federate
Ownership Divestiture Condition:	unconditional
User Supplied Tag	selected by FUT
Federate Name:	FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Request Attribute Ownership Assumption service to the FUT with the supplied parameters:

Object Id:	*supplied by Test Federate*
Attribute Names:	*supplied by Test Federate*

Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall return the parameters:

Attribute Name:	selected from those supplied by RTI

The RTI shall invoke the Attribute Ownership Acquisition Notification to the FUT with the supplied parameters:

Object Id:	*supplied by FUT*
Attribute Names:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Attribute Ownership Divestiture Notification service to Test Federate with the supplied parameters:

Object Ids:	*supplied by FUT*
Attribute Names:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.13 *QUERY ATTRIBUTE OWNERSHIP*

The Query Attribute Ownership function is used to determine if the specified attributes of the specified object Ids are owned and if so by which federate. The STD is shown below.

	SERVICE ISSUED	**RTI STATE**	**FEDERATE STATE**
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	
Event	Federate issues Query Attribute Ownership	Check Registrations for valic federate(name), federation(member), Object(id, attribute name); o error go to operating state	Submit Federate(name), Federation(name), Object(id, attribute name) to RTI
Post Conditions		Return Federate(name) or none	

5.13.1 Traceability

Section 5.7 Query Attribute Ownership

5.13.2 Initiating Query Attribute Ownership

The FUT (as FM) shall invoke the Query Attribute Ownership service with the supplied parameters:

Object Id:	supplied by FUT
Attribute Name:	selected from RID
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Federate Name:	*name or "none"*

5.14 *TIME ADVANCE FUNCTION*

The Time Advance function requests and advance of the federate’s logical time. Invocation of this service implies that the following messages are eligible for delivery to the federate: (1) all incoming receive ordered messages, and (2) all messages using other ordering services with timestamp less than or equal to t. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Federate is internally consistent and ready to move to specified logical time; Subscribed to interaction class and/or Attribute Value
Event 1	Federate issues Time Advance Request	Check Registrations for Federate(name), Federation(member), check for valid Time(logical); on error go to operating state	Submit Federate(name), Federation(name), Time(logical) to RTI
Post Conditions		If events eligible go to E2 & deliver message to federate; if no eligible messages, go t E3	
Event 2	RTI issues Receive Interaction (or Reflect Attribute)	Submit Interaction(class name, parameter names, values), and Time(federation tag, handle) to the federate	Check RID for valid Interaction(class, names, parameters); check Registrations for subscribed Interactions; on error go to operating state
Post Conditions		For TSO: Time(logical) = Time(federation); if non-TSO: Time(logical) = Time(logical)	
Event 3	RTI issues Time Advance Grant	Submit Time(federation) to federate	Check Registrations for valid Time(logical); on error go to operating state
Post Conditions		Register Time(logical) for Federaten(name)	

5.14.1 Traceability

Section 6.7 Time Advance Request
Section 6.10 Time Advance Grant

Section 4.4 Discover Object
Section 4.9 Remove Object
Section 4.5 Reflect Attribute Values
Section 4.7 Receive Interaction

5.14.2 Initiating Time Advance

The FUT shall invoke the Time Advance Request service with the supplied parameters:

Logical Time:	supplied by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Receive Interaction service to the FUT with the parameters:

Interaction Class Name:	*supplied by Test Federate*
Interaction Parameter Names, Values:	*supplied by Test Federate*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*supplied by FUT*
Retraction Handle:	*supplied by RTI*
Federate Handle:	*supplied by RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Time Advance Grant service with the supplied parameters:

Federation Time:	*supplied by RTI*
Federate Handle:	*supplied by RTI*
Federation Execution Name:	*Test*

5.14.3 Responding to Time Advance

No separate test required; covered in 6.9.2.

5.15 *NEXT EVENT*

The Next Event function is used to request the next time stamp ordered (TSO) message from the RTI, provided that message has a time stamp no greater than the logical time specified in the request. The invocation of this service implies that the following messages are eligible for delivery to the federate: (1) all receive ordered messages, (2) the smallest time stamped TSO message that will ever be delivered in the future with time stamp less than or equal to the specified time, and all other TSO messages containing this same time stamp value, and (3) all messages using other ordering services with time stamp less than or equal to the specified time. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Federate is internally consistent and ready to move to specified logical time; Subscribed to interaction class and/or Attribute Value
Event 1	Federate issues Next Event Request	Check for valid Time(logical); on error go to operating state	Submit Time(logical, #events) to RTI
Post Conditions		If events eligible go to E2 & deliver message to federate; if no eligible messages, go t E3	
Event 2	RTI issues Receive Interaction (or Reflect Attribute)	Submit Federate(name), Federation(name), Interaction(class, parameter names, values), and Time(federation, tag, handle to the federate	Check RID for valid Interaction(class names, parameters); check Registrations for subscribed Interactions; on error go to operating state
Post Conditions		For TSO: Time(logical) = Time(federation); if non-TSO: Time(logical) = Time(logical)	
Event 3	RTI issues Time Advance Grant	Submit Time(federation) to federate	Check Registrations for valid Time(logical); on error go to operating state
Post Conditions		Register Time(logical) for Federaten(name)	

5.15.1 Traceability

Section 6.8 Next Event Request
Section 6.10 Time Advance Grant
Section 4.4 Discover Object
Section 4.9 Remove Object
Section 4.5 Reflect Attribute Value
Section 4.7 Receive Interaction

5.15.2 Initiating Next Event

The FUT shall invoke the Next Event Request service with the supplied parameters:

Federate Time:	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Receive Interaction service to the FUT with the parameters:

Interaction Class Name:	*supplied by Test Federate*
Interaction Parameter Names, Values:	*supplied by Test Federate*
Federation Time:	*supplied by Test Federate*
User Supplied Tag	*supplied by FUT*
Retraction Handle:	*supplied by RTI*
Federate Handle:	*supplied by RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Time Advance Grant service with the supplied parameters:

Federation Time:	*supplied by RTI*
Federate Handle:	*supplied by RTI*
Federation Execution Name:	*Test*

5.15.3 Responding to Next Event

No separate test required; covered in 6.10.2.

5.16 *RETRACT*

The Retract function is used to retract a previously scheduled event. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	Federate has previously issued Update Attribute Values and Send Interaction service calls and obtained retraction handles
Event 1	Federate issues Retract	Check Registrations for valid federate(name), federation(member), and Time(handles); on error go to operating state	Submit Federate(name), Federation(name), Time(handles) to RTI
Post Conditions		RTI checks buffers for retracted event; if buffered, retract and go to operating state, else go to E2	
Event 2	RTI issues Reflect Retraction	Submit Time(handle) to federates	Check Registrations for valid Time(handles)
Post Conditions			Cancel effects of Interaction(handle)

5.16.1 Traceability

Section 4.16 Retract
Section 4.17 Reflect Retraction
Section 4.3 Update Attribute Values
Section 4.6 Send Interaction

5.16.2 Initiating Retract

The FUT shall invoke the Retract service with the supplied parameters:

Retraction Handle:	supplied by RTI in 4.3 (Update Attribute Values) or 4.6 (Send Interaction)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Reflect Retraction service to the Test Federate with the supplied parameters:

Retraction Handle:	*supplied by FUT*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

5.16.3 Responding to Retract

The Test Federate shall invoke the Retract service with the supplied parameters:

Retraction Handle:	supplied by RTI in4.3 (Update Attribute Values) or 4.6 (Send Interaction)
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Reflect Retraction service to the FUT with the supplied parameters:

Retraction Handle:	*supplied by Test Federate*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

6. MANAGEMENT SERVICES

Management services are used to control the operation of the federation or to request information about the state of a federation.

6.1 *PAUSE FEDERATION EXECUTION*

The Pause Federation Execution function is used to stop the advance of the federation. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STATE
Pre-Conditions		Knows federation membership	Member of Federation	Member of Federation
Operating State		Knows services issued	Federation not paused	Federation not paused
Event 1	Federation Manager issues Request Pause	Check Registrations for Federate(name), Federation(member), Pause(pausing); on error go to operating state	Submit Federate(name), Federation(name), Pause(label) to RTI	
Post Conditions		Register Pause(label); previous Pause invalidated by new registration	Federate is running	
Event 2	RTI issues Initiate Pause to Federate B	Submit Pause(label) to federate		Check Registrations for Pause(pausing); on error go to operating state
Post Conditions				Start Pause(pausing); previous Pause invalidated by new registration
Event 3	Federate B issues Pause Achieved	Register Pause(label);on error go to operating state;		Submit Pause(label) to RTI
Post Conditions				Register Pause(running)

6.1.1 Traceability

Section 2.5 Request Pause
Section 2.6 Initiate Pause
Section 2.7 Pause Achieved

6.1.2 Initiating Pause Federation

The FUT (as FM) shall invoke the Request Pause service with the supplied parameters:

Pause Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Pause service to Test Federate with the supplied parameters:

Pause Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Pause Achieved service with the supplied parameters:

Pause Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.1.3 Responding to Pause Federation

The Test Federate (as FM) shall invoke the Request Pause service with the supplied parameters:

Pause Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Pause service to the FUT with the supplied parameters:

Pause Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Pause Achieved service with the supplied parameters:

Pause Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.2 *RESUME FEDERATION EXECUTION*

The Resume Federation Execution function is used to resume advance of the federation. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STAT
Pre-Conditions		Knows federation membership	Member of federation	Member of federation
Operating State		Knows services issued	Federate is Paused	Federate is Paused
Event 1	Federation Manager issues Request Resume	Check Registrations for Federate(name), Federation(member), Pause(pausing); on error go to operating state	Submit Federate(name), Federation(name) to RT	
Post Conditions				
Event 2	RTI issues Schedule Resume	Submit to the federate		Check Registrations for Pause(pausing); on error go to operating state
Post Conditions				Start Restore
Event 3	Federate issues Resume Achieved	on error go to operating state		Submit to the RTI
Post Conditions				Register Pause(running)

6.2.1 Traceability

Section 2.8 Request Resume
Section 2.9 Initiate Resume
Section 2.10 Resume Achieved

6.2.2 Initiating Resume Federation

The FUT (as FM) shall invoke the Request Resume service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Resume service to Test Federate with the supplied parameters:

Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Resume Achieved service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.2.3 Responding to Resume Federation

The Test Federate (as FM) shall invoke the Request Resume service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Resume service to FUT with the supplied parameters:

Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Resume Achieved service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.3 *Save Federation Execution*

The Save Federation Execution function is used to save the state of the federation. A rolling save takes place as the federation continues to advance. A paused save takes place after the federation has been paused. The STD is shown below.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STAT
Pre-Conditions		Knows federation membership	Member of federation	Member of federation
Operating State		Knows services issued	Knows current state	Knows current state
Event	Federation Manager issues Request Federate Save	Check Registrations for Federate(name) and Federation(member); o error go to operating state	Submit Federate(name), Federation(name) Save(label) to the RTI	
Post Conditions		Register Save(label); cancels previously scheduled saves;		
Event 2	RTI issues Initiate Federate Save to Federate B	Submit Save(label) to federate		on error go to operating state
Post Conditions				
Event 3	Federate B issues Federate Save Begun	Check Registrations for Save(saving); on error go to operating state		Submit to RTI
Post Conditions		Have all federates issued Save Begun? If no, go to E2.		Register Save(saving)
Event 4	Federate B issues Federate Save Achieved	on error go to operating state		Submit Save(achieved) to RTI
Post Conditions				

6.3.1 Traceability

Section 2.11 Request Federation Save

Section 2.12 Initiatet Federate Save

Section 2.13 Federate Save Begun
Section 2.14 Federate Save Achieved

6.3.2 Initiating Save Federation

The FUT (as FM) shall invoke the Request Federation Save service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Federate Save service to Test Federate with the supplied parameters:

Save Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Federate Save Begun service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The Test Federate shall invoke the Federate Save Achieved service with the supplied parameters:

Save Success Indicator:	supplied by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.3.3 Initiating Save Federation (Optional Parameter)

The FUT (as FM) shall invoke the Request Federation Save service with the supplied parameters:

Save Label:	Interface
Save Time	selected by FUT/optional
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Federate Save service to Test Federate with the supplied parameters:

Save Label:	*Interface*
Save Time	*selected by FUT/optional*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Federate Save Begun service with the supplied parameters:

Save Time	selected by FUT/optional
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The Test Federate shall invoke the Federate Save Achieved service with the supplied parameters:

Save Success Indicator:	supplied by Test Federate
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.3.4 Responding to Save Federation

The Test Federate (as FM) shall invoke the Request Federation Save service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Federate Save service to FUT with the supplied parameters:

Save Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Federate Save Begun service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The FUT shall invoke the Federate Save Achieved service with the supplied parameters:

Save Success Indicator:	supplied by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.3.5 Responding to Save Federation (Optional Parameter)

The Test Federate (as FM) shall invoke the Request Federation Save service with the supplied parameters:

Save Label:	Interface
Save Time	selected by Test Federate/optional
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Federate Save service to FUT with the supplied parameters:

Save Label:	*Interface*
Save Time	*selected by Test Federate/optional*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Federate Save Begun service with the supplied parameters:

Save Time	selected by Test Federate/optional
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The FUT shall invoke the Federate Save Achieved service with the supplied parameters:

Save Success Indicator:	supplied by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.4 RESTORE FEDERATION EXECUTION

The Restore Federation Execution function is used to restore the federation to a previously saved state.

	SERVICE	RTI STATE	FEDERATE A STATE	FEDERATE B STAT
Pre-Conditions		Knows federation membership	Member of federation	Member of federation
Operating State		Knows services issued	Knows current state	Knows current state
Event 1	Federation Manager issues Request Restore	Check Registrations for Federate(name), Federation(member); check for valid Save(label); on error go to operating state	Submit Federate(name), Federation(name), Save(label) to RTI	
Post Conditions				
Event 2	RTI issues Initiate Restore to Federate B	Submit Save(label) to federates		Check Registrations for valid Save(label); on err go to operating state
Post Conditions				Register Save(restoring)
Event 3	Federate B issues Restore Achieved	Check Registrations for valid Save(restoring, label) on error go to operating state		Submit Save(Achieved, label) to RTI
Post Conditions				Federate is in state identical to that when Save was issued

6.4.1 Traceability

Section 2.15 Request Restore
Section 2.16 Initiate Restore
Section 2.17 Restore Achieved

6.4.2 Initiating Restore Federation

The FUT (as FM) shall invoke the Request Restore service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Restore service to the Test Federate with the supplied parameters:

Save Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The Test Federate shall invoke the Restore Achieved service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.4.3 Responding to Restore Federation

The Test Federate (as FM) shall invoke the Request Restore service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Initiate Restore service to the FUT with the supplied parameters:

Save Label:	*Interface*
Federate Handle:	*returned from RTI*
Federation Execution Name:	*Test*

The FUT shall invoke the Restore Achieved service with the supplied parameters:

Save Label:	Interface
Federate Handle:	returned from RTI
Federation Execution Name:	Test

6.5 REQUEST FEDERATION TIME

The Request Federation Time function is used to request the current estimate of the federation time. Federation time is the minimum of the Lower Bound Time Stamp (LBTS) and the current value of federate's Logical Time (LT). The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Estimate of Federation time available; waiting for service to be issued	
Event	Federate issues Request Federation Time	Check Registrations for Federate(name) and Federation(member); check that Time(federation) is available; on error go to operating state	Submit Federate(name), Federation(name) to RTI
Post Conditions		Return Time(federation)	

6.5.1 Traceability

Section 6.1 Request Federation Time

6.5.2 Initiating Request Federation Time

The FUT shall invoke the Request Federation Time service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Federate Time:	*(current minimum of LBTS and LT)*

6.6 *Request LBTS*

The LBTS function is used to request the current value of the Lower Bound Time Stamp (LBTS). The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Time(LBTS) is available; waiting for services to be issued	
Event	Federate issues Request LBTS	Check Registrations for Federate(name) and Federation(member);check that Time(LBTS) is availabl on error go to operating state	Submit Federate(name), Federation(name) to RTI
Post Conditions		Return Time(LBTS)	

6.6.1 Traceability

Section 6.2 Request LBTS

6.6.2 Initiating Request LBTS

The FUT shall invoke the Request LBTS service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Federate Time:	*(current value of LBTS)*

6.7 *REQUEST FEDERATE TIME*

The Request Time function is used to request the current value of the federate's Logical Time. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Logical time has been set; waiting for services to be issued	
Event	Federate issues Request Federate Time	Check Registrations for Federate(name) and Federation(member); check that Time(federate) has been set; on error go to operating state	Submit Federate(name), Federation(name) to RTI
Post Conditions		Return Time(federate)	

6.7.1 Traceability

Section 6.3 Request Federate Time

6.7.2 Initiating Request Federate Time

The FUT shall invoke the Request Time service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Federate Time:	*(current value of LT)*

6.8 *REQUEST MINIMUM NEXT EVENT TIME*

The Request Minimum Next Event Time service is used to request the minimum of LBTS and the time stamp of the next Time Stamp Ordered (TSO) message that is held by the RTI for delivery to the requesting federate.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Time(Minimum Next Event Time) is available; waiting f services to be issued	
Event	Federate issues Request Federate Time	Check Registrations for Federate(name) and Federation(member); check that Time(Minimum Next Event Time) is available; on error go to operating state	Submit Federate(name), Federation(name) to RTI
Post Conditions		Return Time(Minimum Next Event Time)	

6.8.1 Traceability

Section 6.4 Request Minimum Next Event Time

6.8.2 Initiating Request Minimum Next Event Time

The FUT shall invoke the Request Minimum Next Event Time service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Federate Time:	*(current minimum of LBTS and the head of the TSO queue)*

6.9 *REQUEST LOOKAHEAD*

The Request Lookahead function is used to request the current value of lookahead for the federate. The current value of the lookahead may differ temporarily from the desired lookahead given in the Set Lookahead service if the value of the lookahead has been reduced. The STD is shown below.

	SERVICE ISSUED	**RTI STATE**	**FEDERATE STATE**
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Federate lookahead has beer set; waiting for services to b issued	
Event	Federate issues Request Lookahead	Check Registrations for Federate(name), Federation(member); check that Time(lookahead) has been set; on error go to operating state	Submit Federate(name), Federation(name) to RTI
Post Conditions		Return Time(lookahead)	

6.9.1 Traceability

Section 6.6 Request Lookahead

6.9.2 Initiating Request Lookahead

The FUT shall invoke the Request Lookahead service with the supplied parameters:

Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall return the parameters:

Lookahead:	*federate's current value of lookahead*

6.10 *FLUSH QUEUE REQUEST*

The Flush Queue Request function releases all messages stored in the RTI's internal queues and delivers them to the federate. The STD is shown below.

	SERVICE ISSUED	RTI STATE	FEDERATE STATE
Pre-Conditions		Knows federation membership and RTI initialization content	Member of Federation
Operating State		Waiting for services to be issued	
Event	Federate issues Flush Queue Request	Check Registrations for Federate(name), Federation(member); check that Time(federation) is vali on error go to operating state	Submit Federate(name), Federation(name) , Time(federation) to RTI
Post Conditions		Check Registrations for pending messages for federate(name); on error go t operating state	

6.10.1 Traceability

Section 6.9 Flush Queue Request
Section 4.7 Receive Interaction
Section 4.10 Time Advance Grant

6.10.2 Initiating Flush Queue Request

The FUT shall invoke the Flush Queue Request service with the supplied parameters:

Federation Time:	selected by FUT
Federate Handle:	returned from RTI
Federation Execution Name:	Test

The RTI shall invoke the Receive Interaction service to the FUT with the parameters:

Interaction Class Name:	*supplied by FUT*
Interaction Parameter Names, Values:	*supplied by FUT*
Federation Time:	*supplied by FUT*
User Supplied Tag	*selected by FUT*
Retraction Handle:	*selected by RTI*
Federate Handle:	*selected by RTI*
Federation Execution Name:	*Test*

The RTI shall invoke the Time Advance Grant service with the supplied parameters:

Federation Time:	*supplied by RTI*
Federate Handle:	*supplied by RTI*

Federation Execution Name:	*Test*

7. APPENDIX A: SERVICE IMPLEMENTATION STATEMENT

SERVICE IMPLEMENTATION STATEMENT

INTERFACE SPECIFICATION V1.0

ITEM	SERVICE	REFERENCE	SUPPORT
FM1	Create Federation Execution	2.1	Yes No
FM2	Destroy Federation Execution	2.2	Yes No
FM3	Join Federation Execution	2.3	Yes No
FM4	Resign Federation Execution	2.4	Yes No
FM5	Request Pause	2.5	Yes No
FM6	Pause Achieved	2.7	Yes No
FM7	Request Resume	2.8	Yes No
FM8	Resume Achieved	2.10	Yes No
FM9	Request Federation Save	2.11	Yes No
FM10	Federate Save Begun	2.13	Yes No
FM11	Federate Save Achieved	2.14	Yes No
FM12	Request Restore	2.15	Yes No
FM13	Restore Achieved	2.17	Yes No
DM1	Publish Object Class	3.1	Yes No
DM2	Publish Interaction Class	3.2	Yes No
DM3	Subscribe Object Class Attribute	3.3	Yes No
DM4	Subscribe Interaction Class	3.4	Yes No
OJM1	Request Id	4.1	Yes No
OJM2	Register Object	4.2	Yes No
OJM3	Update Attribute Values	4.3	Yes No
OJM4	Send Interaction	4.6	Yes No
OJM5	Delete Object	4.8	Yes No
OJM6	Change Attribute Transportation Type	4.10	Yes No
OJM7	Change Attribute Order Type	4.11	Yes No
OJM8	Change Interaction Transportation Type	4.12	Yes No
OJM9	Change Interaction Order Type	4.13	Yes No

OJM10	Request Attribute Value Update	4.14	Yes No
OJM11	Retract	4.16	Yes No
OWM1	Request Attribute Ownership Divestiture	5.1	Yes No
OWM2	Request Attribute Ownership Acquisition	5.5	Yes No
OWM3	Query Attribute Ownership	5.7	Yes No
TM1	Request Federation Time	6.1	Yes No
TM2	Request LBTS	6.2	Yes No
TM3	Request Federate Time	6.3	Yes No
TM4	Request Minimum Next Event Time	6.4	Yes No
TM5	Set Lookahead	6.5	Yes No
TM6	Request Lookahead	6.6	Yes No
TM7	Time Advance Request	6.7	Yes No
TM8	Next Event Request	6.8	Yes No
TM9	Flush Queue Request	6.9	Yes No
DDM1	Create Update Region	7.1	Yes No
DDM2	Create Subscription Region	7.2	Yes No
DDM3	Associate Update Region	7.3	Yes No
DDM4	Modify Region	7.5	Yes No
DDM5	Delete Region	7.6	Yes No

REFERENCES

F. Halsall, Data Communications, Computer Networks and Open Systems, Third Edition, Addison-Wesley Publishing Company, 1992.

B. Zeigler, Theory of Modelling and Simulation, Robert E. Krieger Publishing Company, 1984.

K. Knightson, OSI Protocol Conformance Testing: IS 9646 Explained, McGraw-Hill, Inc., 1993.